The research reported in this paper was sponsored by the Advanced Research Projects Agency Information Processing Techniques Office and was monitored by the Electronic Systems Division, Air Force Systems Command under contract F1962867C0004 with the System Development Corporation.

TECH MEMO *a working paper*

System Development Corporation / 2500 Colorado Avenue / Santa Monica, California 90406
Information International Inc. / 11161 Pico Boulevard / Los Angeles, California 90064

TM-2710/510/00

AUTHOR M. V. Howard, SDC
TECHNICAL J. A. Barnett, SDC
RELEASE L. Hawkinson, III
C. Weissman, SDC
for J. I. Schwartz, SDC

DATE 10/14/66 PAGE 1 OF 5 PAGES

Operating Instructions for the LISP 2 Supervisor in the LISP 2 Core Image

ABSTRACT

This Tech Memo contains operating instructions for the LISP 2 Supervisor in the LISP 2 Core Image.

1. INTRODUCTION

The LISP 2 Supervisor is an interactive aid available in the LISP 2 system for controlling input/output operations and for directing the compilation process.

2. THE *function** LISP

The LISP 2 Supervisor is a *function* named LISP in section LISP. The *dummy:function:declaration* is

```
(FUNCTION((LISP . LISP)NOVALUE)
   ((INFILE SYMBOL)(OUTFILE SYMBOL)(FORMAT SYMBOL)))
```

where INFILE is the name of an opened file from which the Supervisor takes its inputs. OUTFILE is the name of an opened file on which the Supervisor prints the LAP code for *instructions:definitions*, *routine:definitions*, *macro:definitions* and *function:definitions* that are compiled. *Lap:definitions* are duplicated on OUTFILE. The printing is controlled by (PRNLAP . LISP), a FLUID, BOOLEAN variable, and is done only if PRNLAP is TRUE. FORMAT, the third argument, indicates whether the inputs are Intermediate or Source Language, etc. The alternatives are described in Section 4.

*Italicized words are part of the meta-language of the LISP 2 Intermediate Language; see TM-2710/220/01, dated 7 July 1966.

The *function* LISP operates on the specified files until a *terminator* is encountered. A *terminator* is one of the following:

```
STOP
(STOP)
END    (Not to be used with Source Language Input.)
end-of-file character
end-of-media character
```

The Supervisor accepts all *operations* specified in the Intermediate Language document. *Declaratives* are absorbed and *expressions* are evaluated to produce a value. For each *operation* the value or variable name is printed on the user's teletypewriter (file OTTY).

Function:definitions, *routine:definitions*, *macro:definitions* and *instructions: :definitions* are compiled into equivalent *lap:definitions*. *Lap:definitions* are assembled in core if the BOOLEAN, FLUID variable (BINLAP . LISP) is TRUE, otherwise the LAP code is not assembled. This option is useful when compiling for diagnostic purposes only, or when building a library for future use (with PRNLAP set TRUE). If an error is encountered, a diagnostic is output on OTTY and the LAP is discarded. If an error occurs while evaluating an *expression*, the process ceases.

Unlike LISP 1.5, the LISP 2 Supervisor does not quote arguments. *Expressions* at any "level" are evaluated in an identical manner. The value of

```
(CAR(CONS(QUOTE A)(QUOTE B)))
```

is A, not CONS.

When entering the *function* LISP, the message LISPENTRY is output on OTTY. When a *terminator* is input, the message LISPEXIT is output on OTTY and LISP executes a *return:statement*.

3. SUPERVISOR VARIABLES

3.1 THE KEEP LIST

Garbage collection may occur at any time. Variable structures that are created may be reclaimed unless referenced. Therefore, a list of pointers is maintained to prevent garbage collection of variable structures created by *free: :declarations* or *dummy:declarations*. This list of pointers is known as the KEEP list. The KEEP list is updated each time a new *declarative* is processed so that subsequent compilations and evaluations may still reference that declared *variable*. The maintenance of the KEEP list is described in Section 4.

3.2 (GNLIST . SYS)

(GNLIST . SYS), a SYMBOL, FLUID *variable*, is maintained by the Supervisor as an aid to the read program (see description of LISP 2 I/O, TM-2710/240/01 for more information). Section 4 explains the binding points for GNLIST.

4. FORMATS

FORMAT, the third argument of the *function* LISP, specifies the form of the input in INFILE. The four allowable formats are described below.

4.1 ED2

Format ED2 specifies an INFILE consisting of a series of *edlisp:files*. An *edlisp:file* has the following definition:

edlisp:file = (*atom operation**)

where the *operations* are in Intermediate Language.

Two passes are performed on each *edlisp:file*. During the first pass the series of *operations* is scanned and *declarative* information absorbed by the system. During the second pass, *expressions* are evaluated, *variables* in *free:declarations* are preset, and all necessary compilations and assemblies are performed. The two-pass scheme avoids the necessity of using *dummy:declarations* for intra-*edlisp:file* communication. Entrance to LISP with this format causes KEEP and GNLIST to be rebound. The series of *edlisp:files* in INFILE is processed until a *terminator* is input. Note that the degenerate *edlisp:file*, (STOP),is a *terminator*.

4.2 ED1

Use of format ED1 is the same as use of ED2 except that only the first pass (the declarative scan) is performed and neither KEEP nor GNLIST is rebound.

4.3 IL

Format IL specifies that INFILE consists of a series of *operations* written in LISP 2 Intermediate Language. The *operations* are processed, one at a time, until a *terminator* is encountered. Entrance to the *function* LISP with this format causes KEEP and GNLIST to be rebound. This is useful for direct interaction with the system. However, *dummy:declarations* may be necessary.

4.4 SL

A variety of SL formats will be included for working with Source Language. However, as of this date, the syntax translator is not a part of the LISP 2 core image.

5. *Section:name* AND *Default:type*

Whenever LISP is called, the *section:name:list* is initialized to (USER LISP) and the *default:type* to SYMBOL. The *section:name:list* remains unchanged until a *section:declaration* is encountered by LISP. At this point, if the new *section::name:list* does not include LISP, LISP will be attached, e.g., if the Supervisor encounters

```
(SECTION NEW INTEGER)
```

then *section:name:list* becomes (NEW LISP) and *default:type* is INTEGER. If a *default:declaration* is processed, the *default:type* changes without changing the *section:name:list*. Whenever a *terminator* is encountered, LISP exits and restores the *section:name:list* and *default:type* that were active when the *function* LISP was called. FLUID variables are used to maintain *section:name:list* and *default:type*. These variables are bound by each call to LISP. (For more information and examples, see TM-2710/220/01.)

6. SYSTEM INITIALIZATION

When the GO command is given to TSS, the following *form* is evaluated:

```
(LISP(QUOTE ITTY)(QUOTE OTTY)(QUOTE IL))
```

This *function:call* is embodied in a *try:statement* so that any unwrap returns to make a similar call (see Section 7 for more information on *try:statements*). LISP behaves as an ordinary *function* in that it may be called from any place at any time. The normal use of LISP is to allow input of data from a file other than ITTY.

7. AUXILIARY FUNCTIONS

The *functions* described below may be used to aid debugging in the LISP 2 system.

7.1 (FINDEC . LISP)

The *dummy:function:declaration* is:

```
(FUNCTION((FINDEC . LISP)SYMBOL)
     ((NAME SYMBOL)(SECTION SYMBOL)))
```

The two arguments are an *f:name* and a *section:name* that correspond to the two parts of a *tailed:variable*. The value is NIL if a *declaration* does not exist for the *variable*, otherwise the value is a list of (1) the address of the *variable's* triple cell, (2) the contents of the first word of the triple cell, and (3) the *declaration* for the *variable*.

7.2 (EVAL . LISP)

The *dummy:function:declaration* is:

```
(FUNCTION((EVAL . LISP)SYMBOL)
 ((SLIST SYMBOL)(DTYPE SYMBOL)(EXP SYMBOL)))
```

The three arguments are a *section:name:list*, a *default:type* and an Intermediate Language *expression* to be operated. The value of EVAL is the value of EXP evaluated with the given *section:name:list* and *default:type* in effect. Neither KEEP nor GNLIST is rebound.

7.3 (ERROR . LISP)

The *dummy:function:declaration* is:

```
(FUNCTION((ERROR . LISP)SYMBOL)((M SYMBOL)))
```

ERROR prints its argument M on OTTY if the BOOLEAN, FLUID *variable* (PRNERR . LISP) is TRUE. ERROR then calls EXIT with the argument M. All system execution-time errors use this mechanism. PRNERR initially has value TRUE. Compiler diagnostics are printed regardless of the setting of PRNERR. (The compiler does not call ERROR.)

7.4 (EXIT . LISP)

The *dummy:function:declaration* is:

```
(FUNCTION((EXIT . LISP)SYMBOL)((M SYMBOL)))
```

EXIT causes the system to unwrap to the innermost *try:statement*. The argument of EXIT, M, is planted in the locative specified by the *try:statement*. As the unwrap proceeds, the name of each *function* (a dotted pair) is CONS'ed onto the SYMBOL, FLUID *variable* (BACTRC . LISP). The name of the *function* containing the *try:statement* does not appear on this list. If the *try:statement* unwraps to the one embodying the "top-level" call to LISP, the Supervisor sets the SYMBOL, OWN *variable* (BACKTRACE . LISP) to the value of BACTRC. BACTRC is then rebound to NIL. The argument of EXIT is then printed along with the most recursive *function* names on BACKTRACE regardless of the setting of PRNERR. The INTEGER, OWN *variable* (PRNMAX . LISP) specifies the maximum number of *function::names* to be printed.